DM9051A PTP translater

Tom Sun @ DAVICOM

1.Given

從上層傳入 scaled_ppm 到 driver 的 adjust_fine().

2.Rescale

將 scaled_ppm 轉換單位爲 ppb, 並乘上我們的 RATE_BASE (172, 由 2^32/25MHz 得到) DM9051A rate register 是採用這種單位 , 每秒微調 1ns

3. 歷史記錄

board_info 增加一個 s64 欄位 : last_rate, 記錄上次上層 PTP4L 給下來的值 .

4. 計算

算出需要填入 chip 的差值 :
s64 diff_rate = signed_rate - db->last_rate;

5. 正負號分離

將 diff_rate 做正負號分離，準備填入 DM9051A 的 rate register, 無號數是 diff_rate, 負號是 diff_rate_sign:

```
if (diff_rate < 0) { diff_rate = -diff_rate;
        diff_rate_sign = 1;}
```

6. 填入 rate register

```
write_rate_reg(db, diff_rate, diff_rate_sign);
```

```
static int ptp_9051_adjfine(struct ptp_clock_info *ptp, long scaled_ppm)
{
        // 171.798 = 2^32/25M (when rate reg use this value, every sec will increment 1 ns)

        const s32 RATE_BASE = 172;
        struct board_info *db = container_of(ptp, struct board_info,
                                ptp_caps);
#ifdef DE_TIMESTAMP
        printk("+++00112+++++ [in %s] scaled_ppm = %ld, db.last_rate = %lld+++++++++\n",

                __FUNCTION__ ,scaled_ppm, db->last_rate);
#endif

        // per RATE_BASE makes 1 ns increment per 25MHz
        s64 signed_rate = scaled_ppm_to_ppb(scaled_ppm) * RATE_BASE;

        s64 diff_rate = signed_rate - db->last_rate;
        int diff_rate_sign = 0;
        db->last_rate = signed_rate;
        if (diff_rate < 0) {
                diff_rate = -diff_rate;
                diff_rate_sign = 1;
        }

        write_rate_reg(db, diff_rate, diff_rate_sign);
        return 0;
}
```

結論

因爲 DM9051A 需要填入修正差值，而 PTP4L 是給修正值，故上次的修改值就要記錄起來，才能在這次計算差值

Photo by Dave Hoefler on Unsplash